AI-Orchestrated Distributed ML-KEM (CRYSTALS-Kyber) Key Exchange to Mitigate MITM and Backend Key Exposure

A practical, post-quantum, multi-vantage protocol with Fibonacci-timed rekeying for crypto agility and global interception resistance

Author: Paul George Savluc
Date: January 02, 2026
Keywords: post-quantum cryptography, CRYSTALS-Kyber, ML-KEM, MITM prevention, distributed key exchange, key transparency, certificate transparency, Fibonacci key rotation, crypto agility, TLS 1.3, hybrid PQ TLS

Abstract. This paper proposes a practical key establishment design that makes man-in-the-middle (MITM) substitution and single-backend key exposure substantially harder by combining (1) post-quantum key establishment using ML-KEM (standardized from CRYSTALS-Kyber), (2) strong authentication of handshake transcripts, and (3) a distributed, multi-vantage key agreement that derives the final session key from multiple independent KEM secrets obtained through geographically and administratively separated nodes. An AI control plane enforces crypto-agility policies (rotation cadence, node quorum selection, and anomaly-triggered rekeying) without ever gaining direct access to private keys. A Fibonacci-based rekey schedule is introduced as a safe, non-uniform rotation policy that can be clamped to operational limits while preserving unpredictability to external observers. The result is a defense-in-depth blueprint for post-quantum secure sessions compatible with modern transport security (TLS 1.3) and aligned with transparent trust models such as certificate and key transparency.

Contents
1. Motivation and scope
2. Threat model and design goals
3. Background: ML-KEM (CRYSTALS-Kyber) and why authentication matters
4. Protocol: distributed ML-KEM with quorum authentication
5. AI orchestration model (policy without key custody)
6. Fibonacci-timed rotation and time-of-exchange binding
7. Security analysis and limitations
8. Deployment guidance (TLS 1.3 and hybrid post-quantum)
9. Conclusion
References

1. Motivation and scope

Network defenders face two recurring problems during key establishment: (a) active attackers can perform MITM substitution unless the handshake is authenticated, and (b) centralized backends that retain long-lived secrets become high-value targets.
Post-quantum cryptography adds another constraint: systems must resist harvest-now, decrypt-later collection, where traffic is recorded for future decryption by large-scale quantum computers. This paper focuses on a practical architecture that reduces the blast radius of compromise and makes interception-based decryption dramatically less feasible by forcing attackers to defeat multiple independent protections at once. The scope is defensive and operational: implementable protocol structure, cryptographic binding, and deployment patterns that can be incrementally adopted.

2. Threat model and design goals

Adversary classes. We consider (1) passive eavesdroppers capable of recording traffic at one or more network vantage points, (2) active MITM adversaries capable of intercepting and modifying traffic, including localized BGP and routing manipulation, and (3) partial infrastructure compromise, where one region, cloud account, or service component may be penetrated.
Out of scope. Physical compromise of endpoints, malware on client devices, and total compromise of all independent nodes simultaneously are not solvable by protocol design alone.
Design goals.

- Authentication: prevent silent MITM key substitution by binding identity to the handshake transcript.
- Post-quantum secrecy: use ML-KEM for session secrets that remain confidential against quantum-capable adversaries.
- Compartmentalization: avoid a single backend holding enough material to decrypt sessions.
- Forward secrecy and agility: short-lived session keys, frequent rekey, and rapid response to anomalies.
- Multi-vantage hardness: require attackers to observe or compromise multiple independent paths to recover a session key.

3. Background: ML-KEM (CRYSTALS-Kyber) and why authentication matters

ML-KEM (from CRYSTALS-Kyber). ML-KEM is a key encapsulation mechanism (KEM). A server publishes a public key. A client creates a ciphertext and an associated shared secret using encapsulation, transmits the ciphertext, and the server recovers the same shared secret using decapsulation with its private key. Passive observers who record the public key and ciphertext should not be able to recover the shared secret.

Why KEM is not enough. Any unauthenticated key exchange, including KEM-only handshakes, is vulnerable to active MITM. An attacker can replace public keys in transit, negotiate two separate secrets (one with each party), and translate traffic. Therefore, the handshake transcript must be authenticated using a digital signature, a certificate chain

with pinning, or an external quorum of notaries and transparency logs.

Transport context. TLS 1.3 defines a modern handshake structure designed to prevent eavesdropping, tampering, and message forgery, and it provides a natural integration point for post-quantum key establishment and hybrid schemes.

4. Protocol: distributed ML-KEM with quorum authentication

Core idea. Establish multiple independent ML-KEM shared secrets across distinct nodes (different regions, providers, or administrative domains). Derive the final session key from the concatenation of these secrets using HKDF, with transcript binding. A passive eavesdropper must recover all component secrets to decrypt. A partial compromise leaks, at most, one component secret and does not yield the final key.
Entities. Client C and Service S (the real endpoint). Global nodes N1..Nk participate in ML-KEM decapsulation and are separated by geography and administration. An Authenticator A binds node public keys to the service identity (certificates with pinning, signatures, and/or transparency logs).

Step	Action	Security intent
0	Client obtains service identity material (pinned key, certificate chain, or transparency-backed key binding).	Prevents MITM identity substitution.
1	Client requests a node set N1..Nk based on policy (provider diversity, geography, latency).	Creates multi-vantage hardness.
2	Each node Ni provides ML-KEM public key pk_i plus an identity binding (signature, certificate, or quorum proof).	Authenticates pk_i as authorized for S.
3	Client encapsulates to each pk_i: (ct_i, ss_i) = Encaps(pk_i); sends ct_i to node Ni.	Produces independent post-quantum secrets.
4	Each node decapsulates ss_i = Decaps(sk_i, ct_i). Nodes forward sealed shares to S or act as S delegates.	Allows S to compute component secrets without exporting private keys.
5	Client derives final key: K_final = HKDF(ss_1 || ... || ss_k, transcript_hash, context).	Combines secrets and binds to transcript to stop replay and mix-and-match.
6	Key confirmation: both sides exchange MAC(K_final, transcript_hash) to prove agreement.	Detects active interference and mismatch.

Notes on node trust. Nodes should not be able to unilaterally decrypt traffic. A recommended design is share forwarding, where nodes help S derive component secrets but never see final traffic keys, and where node private keys are isolated in HSMs or enclaves. The final session traffic encryption is end-to-end between C and S.
Quorum and transparency. Instead of trusting a single certificate authority or a single region, the client can require a quorum of independent observations for the service identity and node keys. Certificate transparency and key transparency systems provide append-only audit logs that make short-lived substitution attacks easier to detect.

5. AI orchestration model (policy without key custody)

AI is treated as a policy and monitoring control plane, not as a cryptographic principal that holds secrets. The AI is permitted to request operations, but it must not be able to extract private keys or raw session material.
Safe authority boundary.

- Keys are generated inside HSMs or enclaves and marked non-exportable.
- AI can trigger: rotation, revocation, node reweighting, and quarantine actions.
- AI can validate: attestation evidence, certificate and transparency anomalies, path changes, and time skew.
- AI cannot: read private keys, export session keys, or bypass handshake confirmation.

Why this matters. If an AI subsystem can export keys, it becomes the universal backend, and compromise of that one subsystem collapses the entire trust model. The correct pattern is AI as scheduler, HSM as vault.

6. Fibonacci-timed rotation and time-of-exchange binding

Fibonacci timing as controlled non-uniform churn. Rekeying too slowly increases exposure to long-term capture and cryptanalytic improvements. Rekeying too quickly can overload systems and create failure modes. A Fibonacci-based schedule provides a simple non-uniform cadence that is easy to implement and can be clamped to operational bounds.

Let F(1)=1, F(2)=2, F(n)=F(n-1)+F(n-2). Define rotation interval:

interval(n) = clamp(F(n) * T0, min_interval, max_interval)

where T0 is a base time unit (for example 30 seconds). The schedule can reset on risk events, such as route anomalies, certificate surprises, or node compromise alerts.

Time-of-exchange binding. The handshake transcript should include a client nonce, a timestamp, a monotonic counter, the chosen node set, and hashes of each ML-KEM ciphertext. This transcript is authenticated and used as HKDF salt. Recorded handshakes cannot be replayed to force key reuse, and mix-and-match attacks are detected by key confirmation.

7. Security analysis and limitations

Passive interception. An eavesdropper who records pk_i and ct_i for each node cannot feasibly compute ss_i without breaking ML-KEM. If the final key is derived from multiple ss_i values, the attacker must recover all of them.
Active MITM. The design relies on authenticated bindings for node keys and transcript authentication. Without authentication, any key exchange can be MITM-substituted. With transcript authentication plus key confirmation, active tampering produces a mismatch and the session fails closed.
Partial compromise. Compromise of one node (or one cloud region) reveals at most one component secret. The session remains confidential if at least one component secret remains unknown to the attacker. This is an all-or-nothing combiner pattern when the

KDF input concatenates independent secrets.
Limitations.

- If the endpoint device is compromised, protocol guarantees do not hold.
- If an attacker compromises all nodes used for a given session at the time of the handshake, confidentiality collapses.
- Operational complexity increases with k. Choose k based on realistic attack models and latency budgets.

8. Deployment guidance (TLS 1.3 and hybrid post-quantum)

Preferred integration. Deploy inside TLS 1.3 so you inherit mature transcript binding, downgrade protections, and a well-studied key schedule. Post-quantum key establishment can be integrated via ML-KEM TLS named groups and, where policy demands, hybrid ECDHE plus ML-KEM groups to hedge against cryptanalytic surprises.

Identity robustness. Use certificate transparency for CA-issued certificates and consider key transparency or notary systems to detect localized substitution attacks. Multi-path probing and notary observations can reduce the feasibility of localized MITM.

Implementation notes. Use well-maintained libraries for ML-KEM and signatures (for example Open Quantum Safe for prototyping) and keep key material inside HSMs or enclaves in production. Ensure random number generation is robust and hardware-backed where possible.

9. Conclusion

A post-quantum KEM such as ML-KEM (CRYSTALS-Kyber) protects session secrecy against passive collection and future quantum adversaries, but it does not by itself prevent active MITM substitution. This architecture makes MITM and backend key exposure significantly harder by combining authenticated transcripts, multi-vantage distributed KEM secrets, HKDF-based combining, and short-lived rekeying orchestrated by an AI policy layer that never holds keys. The blueprint is intended as a practical path toward crypto-agile, globally resilient key establishment that can be deployed incrementally within TLS 1.3 and audited using transparency and notary techniques.

Signed,
Paul George Savluc

References

1. National Institute of Standards and Technology (NIST). FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM). Aug 13, 2024.

2. National Institute of Standards and Technology (NIST). FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA). Aug 13, 2024.

3. E. Rescorla. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3. IETF, Aug 2018.

4. H. Krawczyk and P. Eronen. RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF). IETF, May 2010.

5. B. Laurie, A. Langley, and E. Kasper. RFC 6962: Certificate Transparency. IETF, Jun 2013.

6. D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. USENIX ATC, 2008.

7. M. S. Melara et al. CONIKS: Bringing Key Transparency to End Users. USENIX Security Symposium, 2015.

8. IETF TLS Working Group. ML-KEM Post-Quantum Key Agreement for TLS 1.3. Internet-Draft draft-ietf-tls-mlkem (active draft).

9. IETF TLS Working Group. Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLS 1.3. Internet-Draft draft-ietf-tls-ecdhe-mlkem (active draft).

10. Open Quantum Safe. liboqs and liboqs-python documentation and APIs for KEM and signature prototyping.